Quarta Lista de Exercícios

Carlos Camarão

Data de Entrega: 7 de Julho de 2009

Para facilitar o trabalho do monitor, por favor escreva as soluções em arquivos `Problema1.java`, `Problema2.java` e `Problema3.java`, com as classes que contêm o mťodo `main` com o mesmo nome do arquivo, sem a extensão `.java`.

1. Reescreva o exercício 3 da 1ª lista de modo a fazer *validação dos dados de entrada*, isto é, de modo a garantir que os dados de entrada sejam sempre valores válidos para o tipo esperado (no caso, valores válidos do tipo `double`).

   Para isso, defina um método para leitura de um valor de tipo `double` que faça uma chamada ao método `parseFloat` da classe `Float` e defina um tratador que i) trate a exceção que pode ser causada pela chamada a esse método (da classe `NumberFormatException`) e ii) chame o método recursivamente.

2. Reescreva o exercício 9 da 1ª lista de modo a garantir que o valor lido seja um valor correto (entre 0 e 100) para a nota de um aluno. O programa deve definir um método para leitura da nota e propagar uma exceção de tipo `NotaInvalida` caso o valor lido não constitua um valor inteiro entre 0 e 100. A classe `NotaInvalida` deve ser definida como uma subclasse da classe `Exception`. O programa deve tratar possível exceção que possa ser causada, de modo a informar ao usuário que o valor de entrada não constitui um valor correto para uma nota.

3. Em Java, uma exceção não é causada quando o resultado de uma operaçõ aritmética com valores inteiros tem magnitude grande demais para poder ser armazenado no espaço de memória reservado para valores desses tipos (essa condição é comumente chamada, em inglês, de *overflow*; em português podemos chamar essa condição de "incapacidade de armazenamento", ou "estouro").

   No caso da adição de dois inteiros, tal condição ocorre se e somente se as duas parcelas da adição têm o mesmo sinal e o sinal do resultado é diferente do sinal das parcelas (quando as duas parcelas têm sinais diferentes entre si, nunca pode ocorrer uma incapacidade de armazemento). Em outras palavras, em caso de incapacidade de armazenamento o resultado é negativo se os operandos são positivos, e positivo se os operandos são negativos .

   Com base nisso, escreva um programa que defina um método estático `soma` que, ao receber dois argumentos do tipo `int`, retorna a soma desses argumentos se não houver incapacidade de armazenamento e causa uma exceção de tipo `ArithmeticException` caso contrário. O programa deve ler repetidamente dois números inteiros diferentes

de zero, chamar o método `soma` para somar os dois inteiros lidos e imprimir, dependendo dos valores dos argumentos, o resultado da soma, caso o valor da soma possa ser armazenado em um valor de tipo `int`, caso contrário uma mensagem apropriada para o usuário (indicando que o valor da soma não pode ser armazenado em um valor de tipo `int`). A leitura deve terminar quando um dos valores lidos for igual a zero.